

Task# 1

IMPORTANT:

THIS TASK MUST BE COMPLETED EXCLUSIVELY BY YOU. NO EXTERNAL HELP OF ANY KIND IS ALLOWED.

IN CASE THESE REQUIREMENTS ARE NOT MET, WE RESERVE THE RIGHT TO WITHHOLD PAYMENT.

Stack Requirements:

- Node.js v22/ ExpressJS
- Angular v19 /Angular Material
- MongoDB

Initial/Setup:

- Create a backend Node.js/ExpressJS Server
- Server should be running on Port 3000 Create a folder:
 - o controllers/
 - o routes/
 - o helpers/
 - o models/
- app.js file should contain all the initializations
- Create an Angular App with Angular Material
- The app should be configured to run on port 4200

Objective:

1. In the Angular Application, create a routing and modules for integration.
2. Create components for the integration.
3. We want to connect to GitHub via API using OAuth (2) authentication.
 - a. Documentation: <https://docs.github.com/en/rest?apiVersion=2022-11-28>.
 - b. When the Connect button is clicked, we will redirect the user to GitHub for Authentication using OAuth (2)
 - c. After the successful integration, we display the success status.
 - d. We will store the authentication details in a MongoDB DB integrations and collection: github-integration

- e. When we refresh the page, if we have already connected the page should have the green checkmark and it should display the date when the integration was connected.
- f. • Store the information of the user who authenticated.
- 4. When we expand the Mat Panel, we should have the following options:
 - a. Remove Integration
 - i. When the Remove button is clicked, it will remove the connection and delete it from the database.
 - b. Re-sync Integration
- 5. We need to fetch the following data from github:
 - a. Organizations
 - b. Organizations/repos
 - c. Organizations/repos/commits
 - d. Organizations/repos/pulls
 - e. Organizations/repos/issues
 - f. Organizations/repos/issues/changelogs
 - g. Organizations/users

Notes:

- For testing purposes, ensure you have atleast 1 organizations within your github account.
- Within your testing organization, make sure you have atleast 3 repos and each repo should have at least 2000 commits. Ensure to make at least 5 pull requests and 5 issues. Ensure your code could handle thousands of pull requests and issues if required. You can import public open source libraries into your test repo as well.
- We will store the above data into their separate collection in the database.
- We need the ability to view all the github data.

Raw Data

New Grid New Global Search Grid

Active Integrations: github Entity: Processed Entity: processed_commits Search: Delete Grid

	_id	Hash	Branch	Message	Date	Repository Name	Repository Uuid	Author Uuid	Author Name	Pullrequest	URL	Checksum
1	<input type="checkbox"/>	685e9f520be752f9ba2a590	4f6c37c9d57313d61d019489	50660559a93393	fix([angular-devkit/build-angular]) exit dev server when CTRL-C is pressed Since version 4, webpack-dev-server by default will shutdown gracefully. This results in CTRL-C needed to be pressed multiple times to exit the process. See: https://github.com/webpack/webpack-dev-server/issues/706d611a82143f6c602027d43b4b4a7a33a78a-Server.js#L11827 Close #22218	2022-08-27 08:08:30	Angular	90367098	17502226	alan-ajagud	https://github.com/webpack-organization/webpack-dev-server/pull/14685	90367098

Rows Selected 0

Select Tickets that have Time Tracked Against them Deselect Tickets that have Time Tracked Against them Include Entity Column in Time Rec

- Active Integrations Dropdown: Github
- Entity Dropdown: List of Collections in the GitHub database.
- Search: Ability to search keyword in the AG Grid.
- AG Grid Table will display all the fields from the collection. It will get the fields dynamically from the selected collection and display them in the AG Grid.

Some additional requirements:

- Utilize the maximum real estate
- All columns should support filters
- Each field in the collection should be a separate field and a separate column.
- Search should apply search in all the columns.
- Implement Pagination on the backend side
- Sorting and filtering on the backend side
- Create all the columns in AG-Grid dynamically based on data that has JSON and array
- Make sure sorting, filtering and pagination is implemented in AG Grid for every instance.
- Implement global search across all collections.
- Make sure the event loop does not get clogged up.

Testing Tips:

- Create a GitHub account for testing purposes.
- Import Open source/Public Repos.
- Connect the testing account to make sure the authentication is working properly.
- AG Grid filters are working properly. All the fields/columns are displayed correctly.

This task is designed to be interpreted as written. Make your own decisions if anything is ambiguous. Unless absolutely necessary, do not reach out to the team with questions that you can find on your own.